International Journal on Advanced Science Engineering Information Technology

Vol.15 (2025) No. 2
ISSN: 2088-5334

Synchronization of Data Transmission between Edge and Cloud Network in IoT-Based Hydroponic Systems

Eni Dwi Wardihani [a], Helmy [a,*], Ari Sriyanto Nugroho [a], Yusnan Badruzzaman [a], Arif Nursyahid [a], Thomas Agung Setyawan [a], Media Fitri Isma Nugraha [b], Clara Silvia Anggreini [a], Fitri Maharani [a]

[a] *Department of Electrical Engineering, Politeknik Negeri Semarang, Semarang, Indonesia*
[b] *National Research and Innovation Agency, Jakarta, Indonesia*
Corresponding author: [*]helmy@polines.ac.id

***Abstract*—The use of Internet of Things (IoT) systems in hydroponic agriculture aims to enhance the efficiency and effectiveness of controlling plant growth parameters in real-time and automatically. However, current IoT based hydroponic systems heavily depend on internet networks for transmitting data. This reliance becomes problematic when the internet connection is unstable or interrupted, causing monitoring data only to reach edge devices and not the cloud, leading to potential data desynchronization. This study focused on developing a robust data synchronization system between edge devices and cloud computing platforms to address this challenge. The primary goal was to collect the latest data at the edge. Where sensors and local control systems operate is consistently mirrored in the cloud without any loss. The research findings demonstrate that the synchronization system effectively achieves this objective over an 8-day testing period. However, practical constraints, such as the TTGO T-Call ESP32 SIM800L device's transmission limit of 861 data points per operation, were observed. Despite this limitation, the system-maintained reliability, with an average transmission delay of 3 minutes considered acceptable within operational tolerances, ensuring uninterrupted system functionality. This synchronization capability is crucial for hydroponic agriculture, enabling seamless monitoring and control of environmental parameters critical for plant growth. By ensuring data integrity across both edge and cloud systems, growers can make informed decisions promptly, optimize resource utilization, and ultimately improve crop yields in IoT enabled hydroponic setups.**

***Keywords*—Hydroponics; IoT; synchronization; edge-cloud computing.**

Manuscript received 7 Aug. 2024; revised 19 Oct. 2024; accepted 24 Nov. 2024. Date of publication 30 Apr. 2025.

I. INTRODUCTION

Hydroponics is a technique of cultivating plants without using soil as a growing medium (planting, cultivating) [1]. It represents an effective method of plant cultivation aimed at replacing soil media to mitigate land constraints and unpredictable climate change [2]. The hydroponic method enables agricultural systems to be controlled according to desired parameters such as temperature, humidity, nutrients, and pH levels required for plant water management [3]. The management of hydroponic methods is often constrained by time and the need for more accuracy of irregular monitoring, which affects the yield that could be more optimal [4].

To maximize the effectiveness of the hydroponic method, it is innovated with the use of the IoT (Internet of Things) system [5]. Hydroponic management can be monitored in real-time and controlled by how the IoT system can automatically control plant growth parameters [6]. Currently, hydroponics with the IoT system depends on the internet network for data transmission [7]. As a result, if the internet network is unstable or disconnected, the hydroponic parameter monitoring data cannot be sent to the cloud; it is only sent to the edge so that the data becomes unsynchronized. To solve this problem, a data synchronization system between the edge and cloud computing is needed. Thus, the latest data at the edge and cloud computing will always be similar so the monitoring and control system can continue running smoothly.

Previous research proposed a two-level synchronization process that enables the adaptive and accurate distribution of computing tasks based on their latency requirements. The CLEDGE system has the disadvantage that SDN controllers on other hybrid computing models may become failures, which may cause problems [8]. Another research addresses the challenges of traditional cloud-based synchronization by introducing fog computing as the middle layer [9]. The

drawback of the system proposed in the journal is the increased workload on user devices and cloud servers due to differential synchronization methods [10]. Another study compared various scenarios to evaluate a container-based edge computing system for data synchronization applications. When data synchronization is required between edge nodes, scalability, and latency are limited. This is a disadvantage compared to cloud-only solutions beyond the same user threshold and data volume threshold [11]. In addition, some offer analytical performance models to evaluate edge-fog-cloud communication architectures in IoT scenarios [12]. However, real-world performance results may differ from these models, and further refinements may be needed to more accurately model edge-fog-cloud architectures in an IoT context [13].

Although various studies have been conducted to improve synchronization and performance in edge and cloud systems, there are still some areas for improvement, such as the previously described issues of network stability, latency, and resource usage efficiency. Therefore, this research contribution is as follows:

- To implement data synchronization between edge and cloud computing in an Internet of Things-based hydroponic parameter monitoring and control system.
- This upgrade is expected to improve the reliability and efficiency of the hydroponic monitoring system, ensuring that data transmission remains effective despite network instability.

II. Materials and Method

A. Computing

Computing is an algorithm used to find a way to solve problems from input data. The input data in question is an input that comes from outside the system environment [14]. The computing used in this research is Edge Computing and Cloud Computing, both computing are described [15] in Fig. 1.

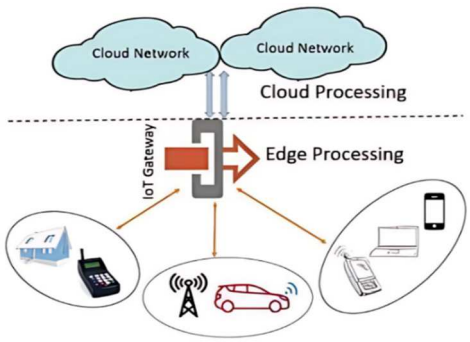


Fig. 1 Edge and Cloud Computing

TABLE I
Main differences between cloud computing and edge computing

Computing	Network	Bandwidth Pressure Network	Mode Real-Time	Calculation Mode
Cloud Computing	Global	More	High	Large-scale centralized processing
Edge Computing	Local	Less	Low	Analysis small-scale intelligent

Cloud computing and edge computing play an important role in the future development of the intelligent Internet of Things [16]. The main differences between cloud computing and edge computing [17] are shown in Table 1.

B. Edge Computing

Edge Computing is an approach to data processing that locates computing resources locally or close to the data-generating device. By processing data near the source, edge computing reduces latency [18], increases bandwidth efficiency, and enhances data security and privacy.

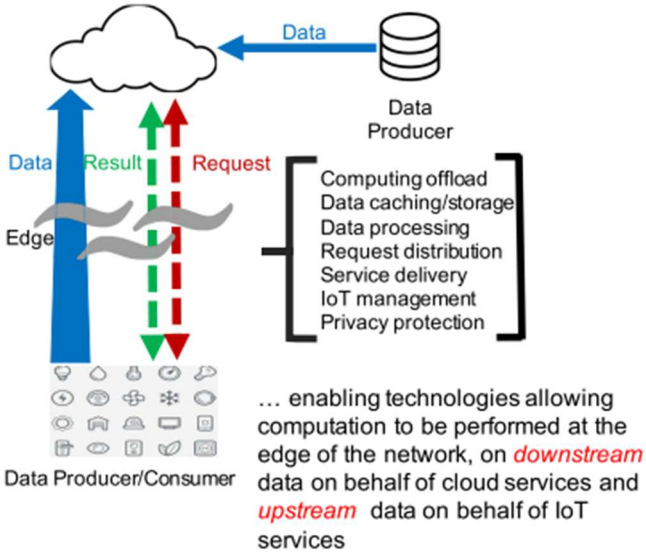


Fig. 2 Illustration of Edge Computing

An illustration of edge computing by [19] is shown in Fig. 2. Edge computing has a two-way computing flow: one direction from the device to the cloud (upstream), and the other direction from the cloud to the device (downstream). This illustrates the bidirectional nature of data flow in edge computing. In this paradigm, end devices not only consume data but also produce it. At the edge, devices can request services and content from the cloud, as well as perform computing tasks uploaded from the cloud. Edge computing can handle tasks such as distributed computing, data storage, caching, processing, request distribution, and delivery of services from the cloud to users [20].

This paradigm supports applications that require rapid response, such as IoT, autonomous vehicles, and health systems, by enabling data processing to be performed largely at the edge of the network, while still collaborating with cloud computing for tasks that require greater computing power or storage. Edge computing also provides flexibility and scalability, and is relevant in various industries such as manufacturing, healthcare, and transportation [21]. In this research, edge computing is used for farmers who are near the equipment so that the internet is not needed if they want to access monitoring data or control hydroponic parameters.

C. Cloud Computing

Cloud computing is a computing model in which computing resources, such as servers, storage, and software, are delivered over the internet [22].

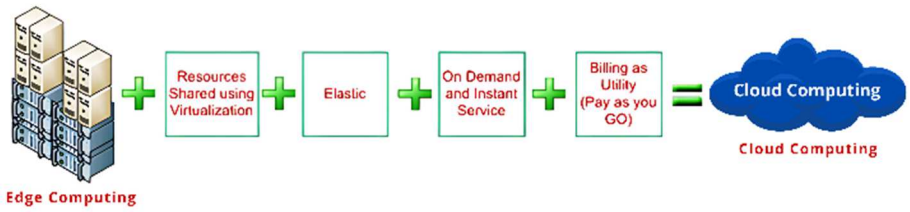


Fig. 3 Schematic Definition of Cloud Computing

The concept of cloud computing refers to a system in which data center resources are shared using virtualization technology which can also provide elastic, on-demand and instant services to customers and allows customers to pay using the pay per use method as shown in Fig. 3 by [23]. Cloud computing provides flexible and scalable IT services without requiring direct maintenance of physical infrastructure [24]. Key advantages include easy access to resources, efficient scalability, and lower costs, with users only paying according to their usage [25]. In this research, cloud computing is used for farmers at home to be able to access the web system it uses the internet network.

D. REST API

Web services are web servers that are built specifically to support the needs of a website or other application [26]. Client programs use Application Programming Interfaces (API) to communicate with web services [27]. In general, an API exposes a set of data and functions to facilitate interaction between computer programs and enable the exchange of information between them [28]. A Web API is the face of a web service, directly listening and responding to client requests [29]. REST APIs are frequently used in web and mobile application development and facilitate system integration by providing a simple, scalable, and easily accessible interface [30]. In this research, REST API is used as a protocol for sending data from tools to the cloud.

E. System Design

The overall system block diagram is shown in Fig. 4, namely a system for monitoring and controlling hydroponic plant cultivation parameters using edge-based machine learning and cloud computing. Several parameters monitored are the degree of acidity (pH), nutrient solution concentration (EC), greenhouse temperature and humidity, nutrient solution level, nutrient solution temperature and light intensity. Meanwhile, the parameters controlled are the acidity of the nutrient solution (pH) and the concentration of the nutrient solution (EC).

With the system below in Fig. 4, farmers can monitor and control hydroponic plant cultivation parameters both online via the web and android with cloud computing technology and offline via android with edge computing technology. From the existing system, an automatic network source switching system will be added which is used for monitoring and control data transmission as well as a synchronization system between data on the edge side and data on the cloud side.

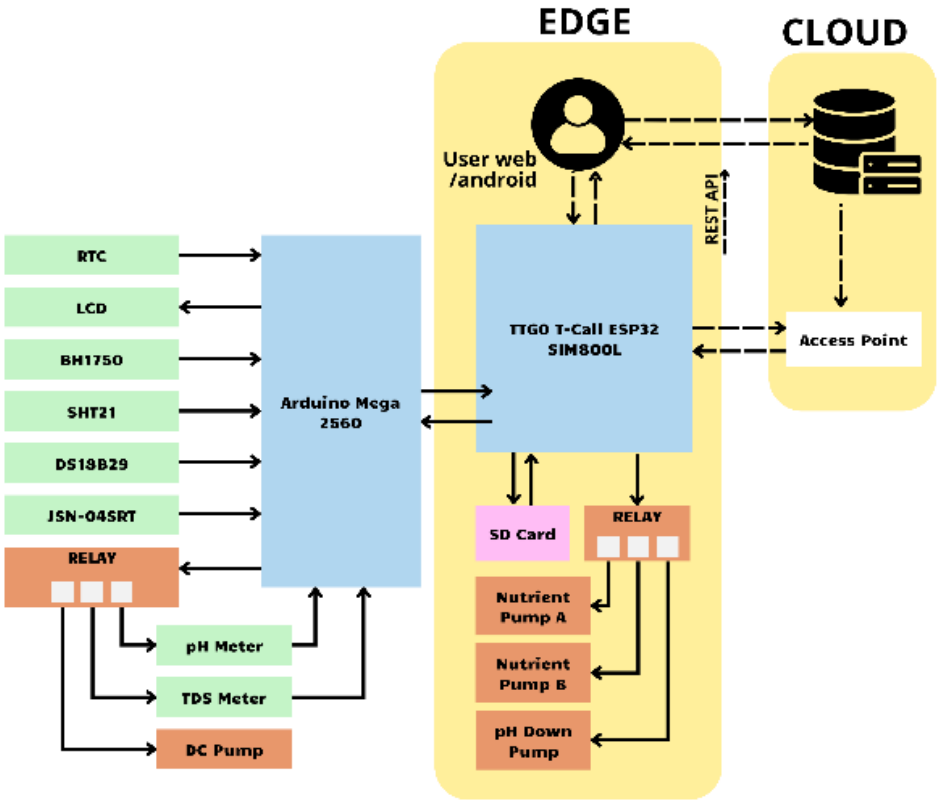


Fig. 4 Block Diagram of Monitoring and Control System

The design of the synchronization system is shown in Fig.4. Synchronization in this system is comparing and synchronizing monitoring data on the SD card with data on the cloud server. The data stored on the SD card is reference data because the data from sensor readings by the Arduino Mega 2560 is directly sent to the TTGO T-Call ESP32 SIM800L module. This module functions to select the best internet network source, so that it can ensure that no data entering the SD card is lost [31].

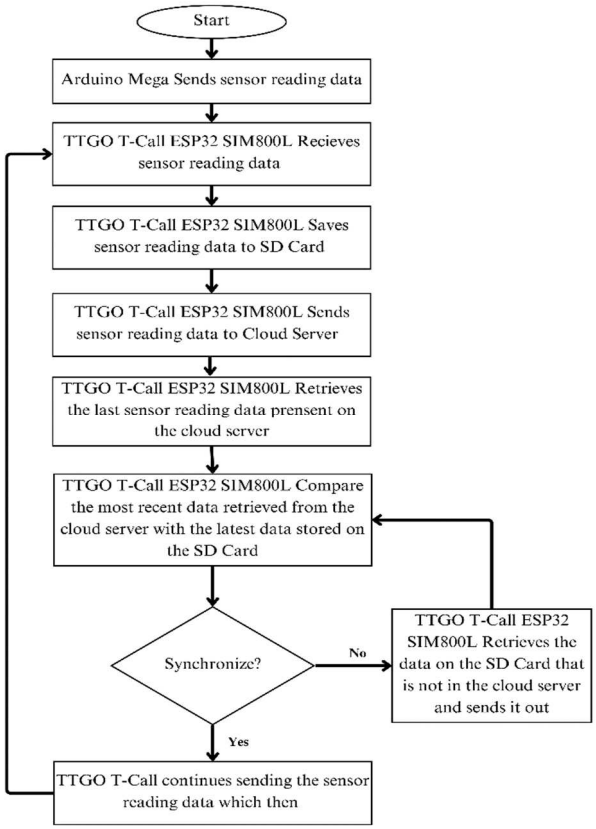


Fig. 5 Synchronization System Flowchart

Fig. 5 is a flowchart for the data synchronization process. This process begins with the Arduino Mega sending sensor readings to the TTGO T-Call ESP32 SIM800L module. Upon receiving the data, the module stores it on a connected SD card. Subsequently, the same data is sent to the cloud server for remote storage. The method used to implement this synchronization system involves the TTGO T-Call ESP32 SIM800L, which retrieves the most recent data entered into the cloud server and compares it with the data stored on the SD card. When the latest data is not synchronized or does not match the last data on the SD card, the TTGO T-Call will send the data that follows the last data on the SD card. If the data entered into the cloud server matches the data stored on the SD card, the synchronization process is complete. This process repeats continuously to ensure that the sensor data remains consistent and secure between local storage and cloud storage.

F. Serial Communication and Data Storage on SD Card

Serial communication in this system occurs between the Arduino Mega 2560 microcontroller and the TTGO T-Call ESP32 SIM800L which is used to send monitoring sensor reading data.

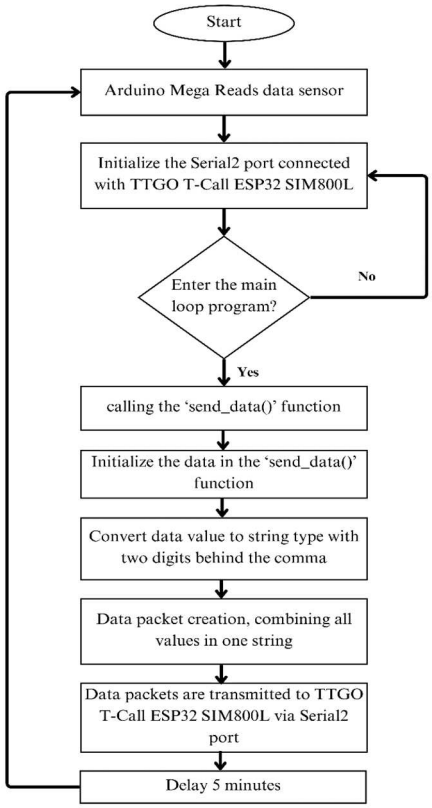


Fig. 6 Serial Communication Flowchart on Arduino Mega a2560

Fig. 6 is a flowchart of the program code on the Arduino Mega 2560 for serial communication with TTGO T-Call ESP32 SIM800L. Serial communication on these two microcontrollers uses the Serial pin on the board.

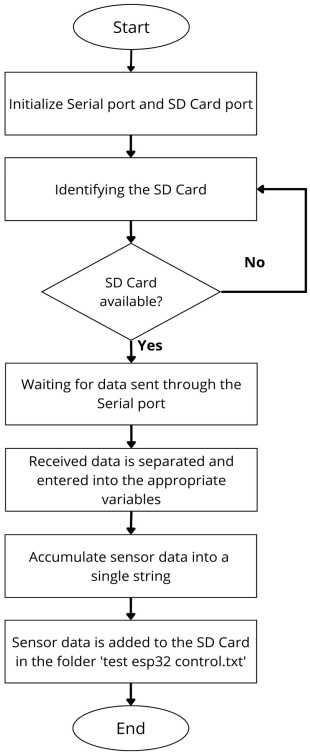


Fig. 7 Edge Data Storage Flowchart on SD Card by TTGO T-Call ESP32 SIM800L

After the data has been successfully sent serially to TTGO T-Call ESP32 SIM800L, then the data will be saved to the connected SD Card. These data are entered in folder *test esp32 control.txt.* The data sent and stored on the SD Card is written in a single line in the file. This line not only includes all the values obtained from the sensors but also includes the date and time information when the readings were taken. To separate each sensor reading value in the line, a semicolon (;) is used as a delimiter. In other words, each line in the text file contains a complete set of sensor data, where each data element is separated by a semicolon, making it easy to parse or read back during the synchronization process. Fig. 7 is the flowchart of the program code for serial communication and data storage on an SD card by TTGO T-Call ESP32 SIM800L.

G. Edge and Cloud Data Synchronization

Data synchronization in this system goes through several stages, namely retrieving the latest data on the server, comparing the data with the latest data on the SD card, and sending data that is not yet on the server.

H. Get Data

The first stage is to retrieve the last data on the server using the HTTP Get method. When the data can be retrieved, the data will be broken down into several variables. This process also changes the server time data to Unix timestamp type. Fig. 8 is a flow diagram for the data get program code carried out by the TTGO T-Call ESP32 SIM800L.

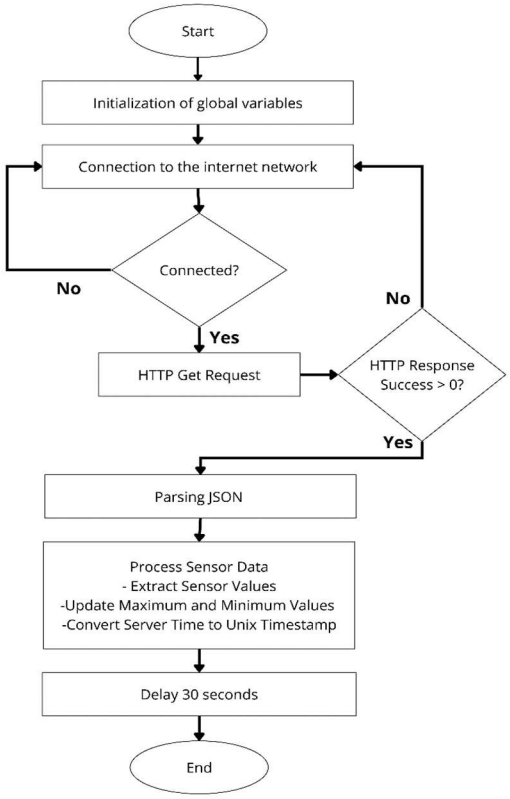


Fig. 8 Flowchart Get Data from Server

I. Compare Data

The next stage is to compare the last data taken by the TTGO T-Call ESP32 SIM800L with the last data on the SD card. The data on the SD card is processed with the buffer command to retrieve date and time data. The date and time data is compared with the Unix timestamp of the last data taken from the server. Fig. 9 is the program code to compare the latest data on the server and SD card.

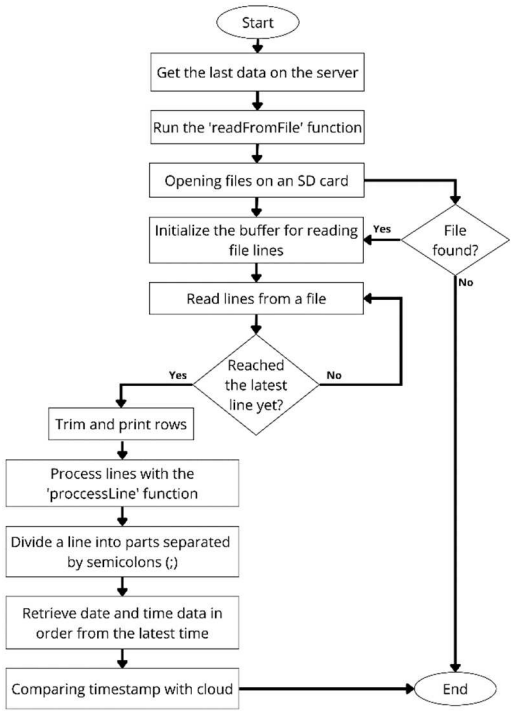


Fig. 9 Compare Data

J. Upload Data

Monitoring data that is not on the server but is on the SD card will be sent after going through the get data and compare data stages. Uploading data on this system uses the HTTPClient protocol and uses the postStr string to send data. Fig. 10 is the program code on the TTGO T-Call ESP32 SIM800L for uploading data.

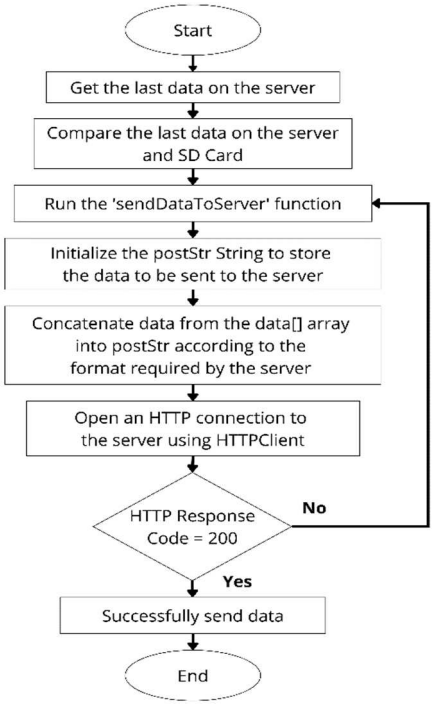


Fig. 10 Upload Data to Server

K. *Synchronization Testing*

Synchronization testing is carried out on the system to obtain the level of accuracy of the running synchronization system [32]. The data from this test consists of the amount of data that was successfully sent from the SD Card to the cloud server and the amount of data that was lost or not successfully sent which is indicated as a system error. To find out the error value, the following formula is used.

$$Error\ (\%) = \frac{\Sigma\ Data\ on\ Edge - \Sigma\ Data\ on\ Cloud}{\Sigma\ Data\ on\ Edge} \times 100 \quad (1)$$

III. Results and Discussion

A. *Synchronization Process Results*

After the sensor reading data is stored on the SD card, the TTGO T-Call ESP32 SIM800L continues the process of retrieving the latest one line of data from the database located on the server omahiot.com. This data is specifically retrieved from the table named hydroponic. An example of the latest one line of data on the database is shown in Fig. 11 The table in Fig. 11 contains 12 columns that store information such as the ID of the greenhouse, sensor readings, time the data was created and sent to the database, and information about the network connected during data transmission to the database. The latest data in the database is monitoring data on 05-06-2024 at 08:46:59.

id	gh_id	ph	tds	level_nutrient	delay	temperature	created_at	updated_at	network	RSSI	HTTP_resp
11289	3	6.01	711.53	57.08	100	28.44	2024-06-05 08:46:59	2024-06-05 08:48:59	wifi	-66	200
11288	3	6.45	752.28	70.91	100	26.94	2024-06-05 08:41:59	2024-06-05 08:43:59	wifi	-69	200
11287	3	6.67	806.72	63.93	100	26.36	2024-06-05 08:36:59	2024-06-05 08:38:58	wifi	-65	200
11286	3	6.33	709.79	81.49	100	26.79	2024-06-05 08:31:59	2024-06-05 08:33:58	wifi	-61	200
11285	3	6.97	778.26	65.12	100	25.67	2024-06-05 08:26:59	2024-06-05 08:28:57	wifi	-68	200
11284	3	6.33	830.99	52.78	100	27.16	2024-06-05 08:21:59	2024-06-05 08:23:56	wifi	-63	200
11283	3	6.03	831.69	67.09	100	28.57	2024-06-05 08:16:59	2024-06-05 08:18:56	wifi	-66	200
11282	3	6.03	723.27	87.29	100	25.40	2024-06-05 08:11:59	2024-06-05 08:13:55	wifi	-66	200
11281	3	6.40	881.65	54.92	100	27.42	2024-06-05 08:06:59	2024-06-05 08:08:55	wifi	-67	200
11280	3	6.72	775.44	78.78	100	28.23	2024-06-05 08:01:59	2024-06-05 08:05:21	wifi	-69	200
11278	3	6.07	852.49	70.73	100	25.58	2024-06-05 07:56:59	2024-06-05 07:58:56	wifi	-60	200

Fig. 11 Latest Data in Database

The process of retrieving data from the database uses the HTTP Get protocol. The successfully retrieved data payload can be seen in the serial monitor on the TTGO T-Call ESP32 SIM800L as shown in Fig. 12 and the latest existing monitoring data retrieved by the microcontroller from the database is data on 05-06-2024 at 08:46:59.

```
09:39:18.578 -> [
09:39:18.578 ->     {
09:39:18.578 ->         "id": "11289",
09:39:18.578 ->         "gh_id": "3",
09:39:18.578 ->         "ph": "6.01",
09:39:18.578 ->         "tds": "711.53",
09:39:18.578 ->         "level_nutrient": "57.08",
09:39:18.625 ->         "delay": "100",
09:39:18.625 ->         "temperature": "28.44",
09:39:18.625 ->         "created_at": "2024-06-05 08:46:59",
09:39:18.625 ->         "updated_at": "2024-06-05 08:48:59",
09:39:18.625 ->         "network": "wifi",
09:39:18.625 ->         "RSSI": "-66",
09:39:18.625 ->         "HTTP_resp": "200"
09:39:18.625 ->     }
09:39:18.625 -> ]
```

Fig. 12 Payload Get Data from Database

TTGO then compares the data just retrieved from the database with the data already stored on the SD card in Fig. 13 The data stored on the SD card is written one line per data, where the contents of one line include the ID of the greenhouse, sensor reading data, and the date and time the data was obtained, separated by a delimiter (;). the data on the SD Card contains sensor reading data on 05-06-2024 at 07:56:59 to 09:01:59. Thus there are three data on the SD Card that are not yet in the database, namely data at 08:51:59, 08:56:59, and at 09:01:59.

```
1  3;6.07;852.49;70.73;100;25.58;2024-06-05;07:56:59;0000-00-00;00:00:00;;;
2  3;6.07;852.49;70.73;100;25.58;2024-06-05;08:01:59;0000-00-00;00:00:00;;;
3  3;6.07;852.49;70.73;100;25.58;2024-06-05;08:06:59;0000-00-00;00:00:00;;;
4  3;6.07;852.49;70.73;100;25.58;2024-06-05;08:11:59;0000-00-00;00:00:00;;;
5  3;6.07;852.49;70.73;100;25.58;2024-06-05;08:16:59;0000-00-00;00:00:00;;;
6  3;6.07;852.49;70.73;100;25.58;2024-06-05;08:21:59;0000-00-00;00:00:00;;;
7  3;6.07;852.49;70.73;100;25.58;2024-06-05;08:26:59;0000-00-00;00:00:00;;;
8  3;6.07;852.49;70.73;100;25.58;2024-06-05;08:31:59;0000-00-00;00:00:00;;;
9  3;6.07;852.49;70.73;100;25.58;2024-06-05;08:36:59;0000-00-00;00:00:00;;;
10 3;6.07;852.49;70.73;100;25.58;2024-06-05;08:41:59;0000-00-00;00:00:00;;;
11 3;6.07;852.49;70.73;100;25.58;2024-06-05;08:46:59;0000-00-00;00:00:00;;;
12 3;6.07;852.49;70.73;100;25.58;2024-06-05;08:51:59;0000-00-00;00:00:00;;;
13 3;6.07;852.49;70.73;100;25.58;2024-06-05;08:56:59;0000-00-00;00:00:00;;;
14 3;6.07;852.49;70.73;100;25.58;2024-06-05;09:01:59;0000-00-00;00:00:00;;;
```

Fig. 13 Data on SD Card

The process of comparing edge and cloud data involves checking the time and date on both data sets to determine which data is more recent as shown in Fig. 14 and there are three new data detected on the SD Card.

```
09:39:23.813 -> Date from SD: 2024-06-05
09:39:23.813 -> Time from SD: 08:51:59
09:39:23.860 -> Timestamp from SD (Unix): 1717577519
09:39:23.860 -> Timestamp from Cloud (Unix): 1717577219
09:39:23.860 -> Data Baru
09:39:23.860 -> ?gh_id=3&ph=6.07&tds=852.49&level_nutrient=70.73&delay=100
09:39:24.001 -> Data successfully sent to server
09:39:24.001 -> Date from SD: 2024-06-05
09:39:24.001 -> Time from SD: 08:56:59
09:39:24.001 -> Timestamp from SD (Unix): 1717577819
09:39:24.001 -> Timestamp from Cloud (Unix): 1717577219
09:39:24.001 -> Data Baru
09:39:24.001 -> ?gh_id=3&ph=6.07&tds=852.49&level_nutrient=70.73&delay=100
09:39:24.237 -> Data successfully sent to server
09:39:24.237 -> Date from SD: 2024-06-05
09:39:24.237 -> Time from SD: 09:01:59
09:39:24.237 -> Timestamp from SD (Unix): 1717578119
09:39:24.237 -> Timestamp from Cloud (Unix): 1717577219
09:39:24.237 -> Data Baru
09:39:24.237 -> ?gh_id=3&ph=6.07&tds=852.49&level_nutrient=70.73&delay=100
09:39:24.381 -> Data successfully sent to server
```

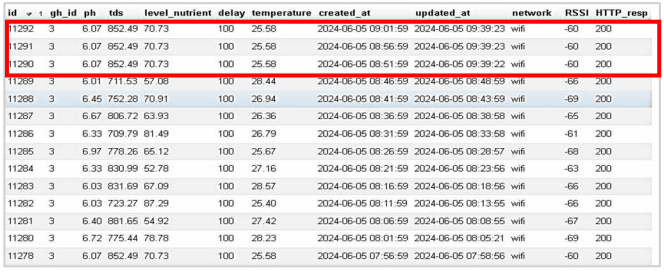
Fig. 14 Data Comparison Process

The time and date on the SD Card and database are converted with the Unix time converter program, which converts time and date into seconds. The Unix time converter program can be seen in Fig. 15.

```
time_t UnixConverter(String date, String time) {
  struct tm timeinfo;
  int year, month, day;
  sscanf(date.c_str(), "%d-%d-%d", &year, &month, &day);
  timeinfo.tm_year = year;
  timeinfo.tm_mon = month;
  timeinfo.tm_mday = day;
  int hour, minute, second;
  sscanf(time.c_str(), "%d:%d:%d", &hour, &minute, &second);
  timeinfo.tm_hour = hour;
  timeinfo.tm_min = minute;
  timeinfo.tm_sec = second;
  time_t unixTime = mktime(&timeinfo);
  return unixTime;
}
```

Fig. 15 Unix Time Converter Program

After the time and date data is converted in Unix time, TTGO T-Call ESP32 will compare the values. If the data timestamp value from the cloud is detected to be smaller than the data timestamp value from the edge, the device will send the newer data to the database. The new data sent to the database can be seen in Fig. 16 that shows three new data detected in the SD Card and sent simultaneously at 09:39:23 to the database.



id	gh_id	ph	tds	level_nutrient	delay	temperature	created_at	updated_at	network	RSSI	HTTP_resp
11292	3	6.07	852.49	70.73	100	25.58	2024-06-05 09:01:59	2024-06-05 09:39:23	wifi	-60	200
11291	3	6.07	852.49	70.73	100	25.58	2024-06-05 08:56:59	2024-06-05 09:39:23	wifi	-60	200
11290	3	6.07	852.49	70.73	100	25.58	2024-06-05 08:51:59	2024-06-05 09:39:22	wifi	-60	200
11289	3	6.01	711.53	57.08	100	28.44	2024-06-05 08:46:59	2024-06-05 08:48:59	wifi	-66	200
11288	3	6.45	752.28	70.91	100	26.94	2024-06-05 08:41:59	2024-06-05 08:43:59	wifi	-69	200
11287	3	6.67	806.72	63.93	100	26.36	2024-06-05 08:36:59	2024-06-05 08:38:58	wifi	-65	200
11286	3	6.33	709.79	81.49	100	26.79	2024-06-05 08:31:59	2024-06-05 08:33:58	wifi	-61	200
11285	3	6.97	778.26	65.12	100	25.67	2024-06-05 08:26:59	2024-06-05 08:28:57	wifi	-68	200
11284	3	6.33	830.99	52.78	100	27.16	2024-06-05 08:21:59	2024-06-05 08:23:56	wifi	-63	200
11283	3	6.03	831.69	67.09	100	28.57	2024-06-05 08:16:59	2024-06-05 08:18:56	wifi	-66	200
11282	3	6.03	723.27	87.29	100	25.40	2024-06-05 08:11:59	2024-06-05 08:13:55	wifi	-66	200
11281	3	6.40	881.65	54.92	100	27.42	2024-06-05 08:06:59	2024-06-05 08:08:55	wifi	-67	200
11280	3	6.72	775.44	78.78	100	28.23	2024-06-05 08:01:59	2024-06-05 08:05:21	wifi	-69	200
11278	3	6.07	852.49	70.73	100	25.58	2024-06-05 07:56:59	2024-06-05 07:58:56	wifi	-60	200

Fig. 16 New Data in Database

B. Synchronization Testing Results

In this test, we want to know how much data is lost or data that does not enter the server during the synchronization process of edge data and cloud data. This test was carried out for 8 days starting on 27/04/2024 to 04/05/2024 in the Hydroponic Greenhouse. The results of synchronization testing can be seen in Table II.

TABLE II
SYNCHRONIZATION TESTING RESULT

Date	Σ Data on Edge	Σ Data on Cloud	Data Difference on Edge and Cloud
4/27/2024	225	225	0
4/28/2024	224	224	0
4/29/2024	224	224	0
4/30/2024	214	214	0
5/1/2024	224	224	0
Total Error (%)	**1769 0**	**1769**	**0**

Table II shows a comparison of the amount of data stored on edge devices and cloud servers for each hour of each day for 8 days. Edge devices are devices that collect data and send it to the cloud server for storage and analysis. The cloud server is a remote data storage. In this test, omahiot.com was used as the cloud server.

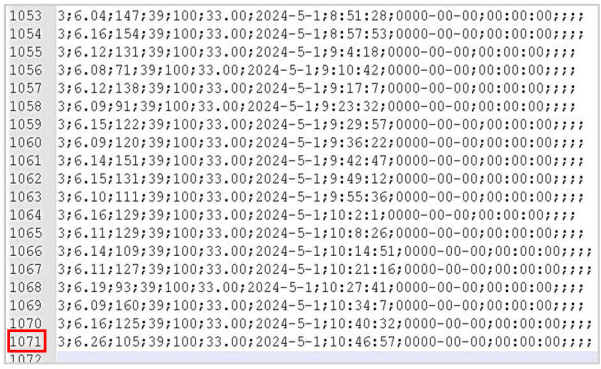
The third column in the test results table shows the total amount of data stored on the edge device every hour. The fourth column shows the total amount of data stored on the cloud server every hour. The fifth column shows the difference in the amount of data stored on the edge device and cloud server every hour. During the testing of this system from 27/04/2024 at 00:00 to 04/05/2024 at 23:59 the total data on the edge and cloud is the same, which is 1769 data. Thus, according to equation (1), an error value of 0% is obtained.

$$Error\ (\%) = \frac{1769-1769}{1769} \times 100\% = 0\% \qquad (1)$$

This is because during testing, the tool is always connected to the internet network properly. There are times when the device is not connected.

C. Synchronization System Error Testing

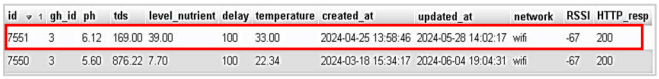
Synchronization Error testing in the synchronization system is done outside of the previous synchronization system testing time. In this test, several data transmission conditions will be made. Then in the fourth test, namely testing the limit of sending data from the edge to the database. This test will see how much data can be sent from the edge to the database. The data transmission limits test result are shown on Fig. 17.



```
1053  3;6.04;147;39;100;33.00;2024-5-1;8:51:28;0000-00-00;00:00:00;;;;
1054  3;6.16;154;39;100;33.00;2024-5-1;8:57:53;0000-00-00;00:00:00;;;;
1055  3;6.12;131;39;100;33.00;2024-5-1;9:4:18;0000-00-00;00:00:00;;;;
1056  3;6.08;71;39;100;33.00;2024-5-1;9:10:42;0000-00-00;00:00:00;;;;
1057  3;6.12;138;39;100;33.00;2024-5-1;9:17:7;0000-00-00;00:00:00;;;;
1058  3;6.09;91;39;100;33.00;2024-5-1;9:23:32;0000-00-00;00:00:00;;;;
1059  3;6.15;122;39;100;33.00;2024-5-1;9:29:57;0000-00-00;00:00:00;;;;
1060  3;6.09;120;39;100;33.00;2024-5-1;9:36:22;0000-00-00;00:00:00;;;;
1061  3;6.14;151;39;100;33.00;2024-5-1;9:42:47;0000-00-00;00:00:00;;;;
1062  3;6.15;131;39;100;33.00;2024-5-1;9:49:12;0000-00-00;00:00:00;;;;
1063  3;6.10;111;39;100;33.00;2024-5-1;9:55:36;0000-00-00;00:00:00;;;;
1064  3;6.16;129;39;100;33.00;2024-5-1;10:2:1;0000-00-00;00:00:00;;;;
1065  3;6.11;129;39;100;33.00;2024-5-1;10:8:26;0000-00-00;00:00:00;;;;
1066  3;6.14;109;39;100;33.00;2024-5-1;10:14:51;0000-00-00;00:00:00;;;;
1067  3;6.11;127;39;100;33.00;2024-5-1;10:21:16;0000-00-00;00:00:00;;;;
1068  3;6.19;93;39;100;33.00;2024-5-1;10:27:41;0000-00-00;00:00:00;;;;
1069  3;6.09;160;39;100;33.00;2024-5-1;10:34:7;0000-00-00;00:00:00;;;;
1070  3;6.16;125;39;100;33.00;2024-5-1;10:40:32;0000-00-00;00:00:00;;;;
1071  3;6.26;105;39;100;33.00;2024-5-1;10:46:57;0000-00-00;00:00:00;;;;
1072
```

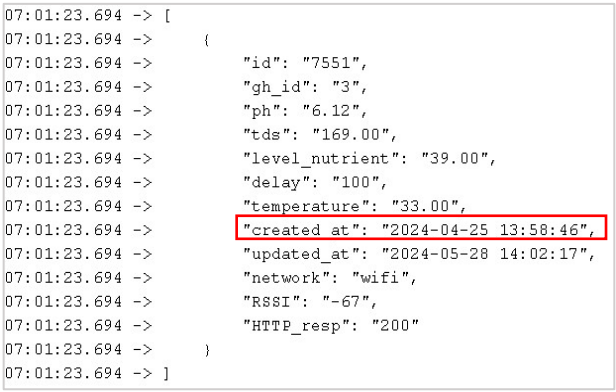
Fig. 17 Dummy data on SD Card

In this data transmission limit test, it is carried out to measure the ability of the tool to compare data and send synchronization data to the database. In this test, we will see how much data can be sent simultaneously at one time. The data used to test be dummy data of 1071 data shown in Fig. 17. The latest data in the database is shown in Fig. 18, which is data on 25-04-2024 at 13:58:46. Then the tool will start the synchronization process by retrieving the latest data in the database. The data payload that was successfully retrieved from the database is shown on Fig. 19.



id	gh_id	ph	tds	level_nutrient	delay	temperature	created_at	updated_at	network	RSSI	HTTP_resp
7551	3	6.12	169.00	39.00	100	33.00	2024-04-25 13:58:46	2024-05-28 14:02:17	wifi	-67	200
7550	3	5.60	876.22	7.70	100	22.34	2024-03-18 15:34:17	2024-06-04 19:04:31	wifi	-67	200

Fig. 18 Dummy data on SD Card



```
07:01:23.694 -> [
07:01:23.694 ->     {
07:01:23.694 ->         "id": "7551",
07:01:23.694 ->         "gh_id": "3",
07:01:23.694 ->         "ph": "6.12",
07:01:23.694 ->         "tds": "169.00",
07:01:23.694 ->         "level_nutrient": "39.00",
07:01:23.694 ->         "delay": "100",
07:01:23.694 ->         "temperature": "33.00",
07:01:23.694 ->         "created_at": "2024-04-25 13:58:46",
07:01:23.694 ->         "updated_at": "2024-05-28 14:02:17",
07:01:23.694 ->         "network": "wifi",
07:01:23.694 ->         "RSSI": "-67",
07:01:23.694 ->         "HTTP_resp": "200"
07:01:23.694 ->     }
07:01:23.694 -> ]
```

Fig. 19 Data Delivery Limit Testing: Last Payload Get Data

Just like the previous test, the next process is to compare the data retrieved from the database with the data on the SD Card per line. The process of comparing and sending data is shown in Fig. 20. When viewed on the serial monitor, it can be seen that the last data compared is not fully read by TTGO, and as a result, the process of comparing the latest data stops, as shown in Fig. 21.

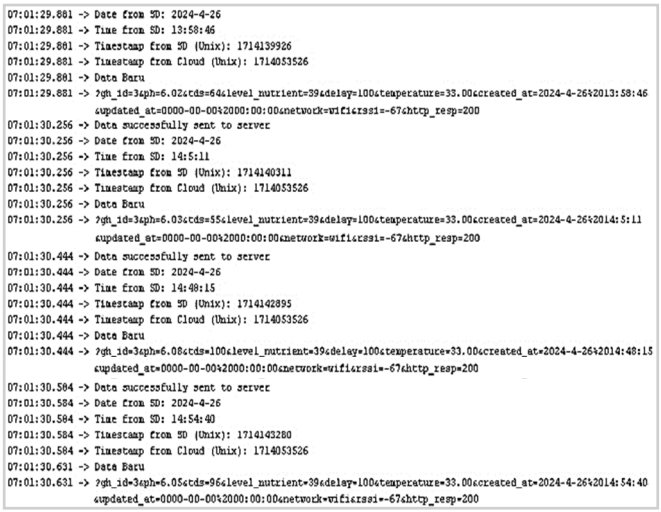


Fig. 20 Data Transmission Limit Testing: Data Synchronization Process

Fig. 21 shows that the synchronization process stopped on 30-04-2024 at 11:09:37. Then it can be seen that in the last data, the monitoring data sent is incomplete, unlike the previous data. Furthermore, when viewed in the database, the last data sent is incomplete, in the "network", "RSSI", and "HTTP_resp" columns contain 0 as shown in Fig. 22.

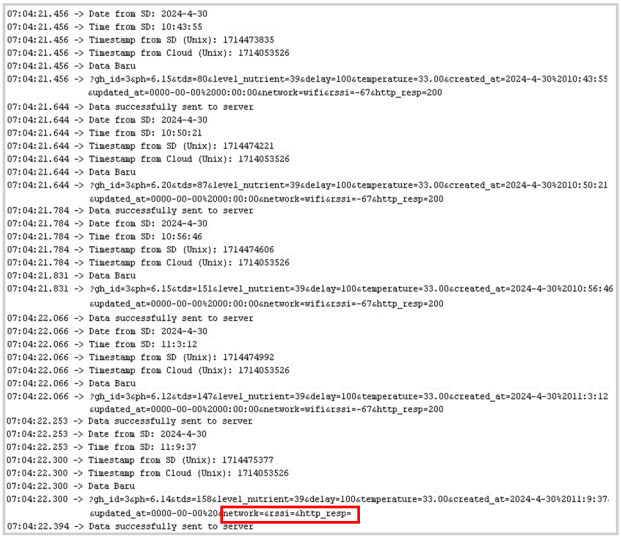


Fig 21 Data Transmission Limit Testing: Data Synchronization Process

id	gh_id	ph	tds	level_nutrient	delay	temperature	created_at	updated_at	network	RSSI	HTTP_resp
11277	3	6.14	158.00	39.00	100	33.00	2024-04-30 11:09:37	2024-06-05 07:04:24		0	0
11276	3	6.12	147.00	39.00	100	33.00	2024-04-30 11:03:12	2024-06-05 07:04:23	wifi	-67	200
11275	3	6.15	151.00	39.00	100	33.00	2024-04-30 10:56:46	2024-06-05 07:04:23	wifi	-67	200
11274	3	6.20	87.00	39.00	100	33.00	2024-04-30 10:50:21	2024-06-05 07:04:23	wifi	-67	200
11273	3	6.15	80.00	39.00	100	33.00	2024-04-30 10:43:55	2024-06-05 07:04:23	wifi	-67	200
11272	3	6.14	151.00	39.00	100	33.00	2024-04-30 10:37:30	2024-06-05 07:04:22	wifi	-67	200
11271	3	6.09	156.00	39.00	100	33.00	2024-04-30 10:31:05	2024-06-05 07:04:22	wifi	-67	200
11270	3	6.18	116.00	39.00	100	33.00	2024-04-30 10:24:39	2024-06-05 07:04:22	wifi	-67	200
11269	3	6.05	181.00	39.00	100	33.00	2024-04-30 10:18:14	2024-06-05 07:04:22	wifi	-67	200
11268	3	6.16	138.00	39.00	100	33.00	2024-04-30 10:11:48	2024-06-05 07:04:22	wifi	-67	200
11267	3	6.09	127.00	39.00	100	33.00	2024-04-30 10:05:23	2024-06-05 07:04:21	wifi	-67	200
11266	3	6.14	174.00	39.00	100	33.00	2024-04-30 09:58:58	2024-06-05 07:04:21	wifi	-67	200
11265	3	6.09	151.00	39.00	100	33.00	2024-04-30 09:52:33	2024-06-05 07:04:21	wifi	-67	200
11264	3	6.08	136.00	39.00	100	33.00	2024-04-30 09:46:07	2024-06-05 07:04:21	wifi	-67	200
11263	3	6.13	176.00	39.00	100	33.00	2024-04-30 09:39:42	2024-06-05 07:04:21	wifi	-67	200
11262	3	6.10	118.00	39.00	100	33.00	2024-04-30 09:33:17	2024-06-05 07:04:21	wifi	-67	200
11261	3	6.11	113.00	39.00	100	33.00	2024-04-30 09:26:52	2024-06-05 07:04:20	wifi	-67	200
11260	3	6.18	111.00	39.00	100	33.00	2024-04-30 09:20:27	2024-06-05 07:04:20	wifi	-67	200

Fig. 22 Data Transmission Limit Testing: Data Synchronization Process

id	gh_id	ph	tds	level_nutrient	delay	temperature	created_at	updated_at	network	RSSI	HTTP_resp
11277	3	6.14	158.00	39.00	100	33.00	2024-04-30 11:09:37	2024-06-05 07:04:24		0	0
11276	3	6.12	147.00	39.00	100	33.00	2024-04-30 11:03:12	2024-06-05 07:04:23	wifi	-67	200
11275	3	6.15	151.00	39.00	100	33.00	2024-04-30 10:56:46	2024-06-05 07:04:23	wifi	-67	200
11274	3	6.20	87.00	39.00	100	33.00	2024-04-30 10:50:21	2024-06-05 07:04:23	wifi	-67	200
11273	3	6.15	80.00	39.00	100	33.00	2024-04-30 10:43:55	2024-06-05 07:04:23	wifi	-67	200
11272	3	6.14	151.00	39.00	100	33.00	2024-04-30 10:37:30	2024-06-05 07:04:22	wifi	-67	200
11271	3	6.09	156.00	39.00	100	33.00	2024-04-30 10:31:05	2024-06-05 07:04:22	wifi	-67	200
11270	3	6.18	116.00	39.00	100	33.00	2024-04-30 10:24:39	2024-06-05 07:04:22	wifi	-67	200
11269	3	6.05	181.00	39.00	100	33.00	2024-04-30 10:18:14	2024-06-05 07:04:22	wifi	-67	200
11268	3	6.16	138.00	39.00	100	33.00	2024-04-30 10:11:48	2024-06-05 07:04:22	wifi	-67	200
11267	3	6.09	127.00	39.00	100	33.00	2024-04-30 10:05:23	2024-06-05 07:04:21	wifi	-67	200
11266	3	6.14	174.00	39.00	100	33.00	2024-04-30 09:58:58	2024-06-05 07:04:21	wifi	-67	200
11265	3	6.09	151.00	39.00	100	33.00	2024-04-30 09:52:33	2024-06-05 07:04:21	wifi	-67	200
11264	3	6.08	136.00	39.00	100	33.00	2024-04-30 09:46:07	2024-06-05 07:04:21	wifi	-67	200
11263	3	6.13	176.00	39.00	100	33.00	2024-04-30 09:39:42	2024-06-05 07:04:21	wifi	-67	200
11262	3	6.10	118.00	39.00	100	33.00	2024-04-30 09:33:17	2024-06-05 07:04:21	wifi	-67	200
11261	3	6.11	113.00	39.00	100	33.00	2024-04-30 09:26:52	2024-06-05 07:04:20	wifi	-67	200
11260	3	6.18	111.00	39.00	100	33.00	2024-04-30 09:20:27	2024-06-05 07:04:20	wifi	-67	200

Fig. 23 Data Transmission Limit Testing: Data Synchronization Process

Then calculating the limit of the amount of data that can be successfully sent simultaneously is the id parameter in the database. It can be seen in Fig. 23 that the first data successfully sent is data with id 10417 and in Fig. 22 the last data that can be sent is data with id 11277. This shows that with the existing system, the maximum limit of data that can be sent by TTGO T-Call ESP32 SIM800L is 861 data. In other words, after reaching this amount, TTGO T-Call ESP32 SIM800L cannot send further data to the database.

D. Delay Testing

This test measures the delay in sending data from the device to the database. Testing was carried out for 8 days in the Hydroponic Greenhouse. The delay test results can be seen in the table and chart below.

TABLE III
DELAY TESTING RESULT

Date	Average Delay (minutes)
4/27/2024	3.83
4/28/2024	3.86
4/29/2024	4.00
4/30/2024	3.93
5/1/2024	3.89
5/2/2024	3.90
5/3/2024	3.90
5/4/2024	3.80

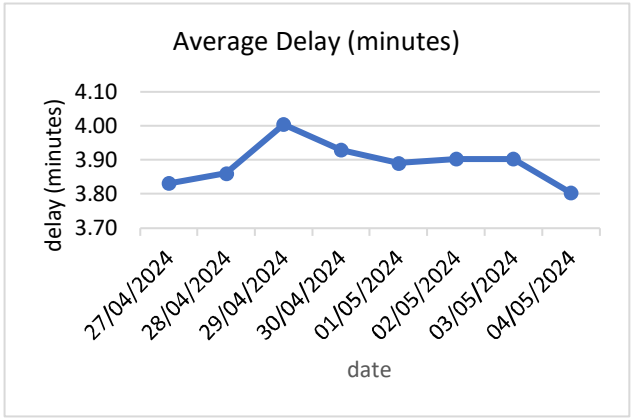


Fig. 24 Delay Testing Results Chart

Fig. 24 is the result of Table III, which shows that the overall average delay test results reached 3 minutes. This is caused by several processes that must be done before the device can send data to the database. However, in operation, this delay does not

interfere with the monitoring and control system. With this delay the system still runs smoothly and does not become an obstacle. In the CLEDGE system, SDN controller failures in the hybrid computing model can compromise network performance and reliability and increase the risk of data corruption. In addition, the application of fog computing as the middle layer also increases the workload on user devices and cloud servers. Therefore, the implementation of the cloud-edge synchronization system effectively ensures that the data at the edge remains synchronized with the cloud without data loss and more efficiently.

IV. CONCLUSION

Based on the research that has been carried out, the following conclusions can be drawn is that the synchronization system effectively ensures that data on the edge is synchronized with the cloud without data loss, as demonstrated by an 8-day testing period, despite the TTGO T-Call ESP32 SIM800L device having a maximum transmission limit of 861 data points at a time. Additionally, delay testing revealed an average delay of 3 minutes, which is considered good and does not disrupt the system's operations.

ACKNOWLEDGMENT

The National Research and Innovation Agency (Badan Riset dan Inovasi Nasional, BRIN) and the Indonesia Endowment Fund for Education are acknowledged for their generous support of this research under the Research and Innovation for Advanced Indonesia (RIIM) scheme.

REFERENCES

[1] E. S. Kaseng, M. A. Syifani, and A. M. A. Mukhlis, "Development and working test of microcontroller-based automatic seedling tools for hydroponic systems," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 14, no. 2, pp. 738-748, Apr. 2024, doi: 10.18517/ijaseit.14.2.19862.

[2] N. K. Bharti et al., "Hydroponics system for soilless farming integrated with Android application by Internet of Things and MQTT broker," in *Proc. IEEE Pune Sect. Int. Conf. (PuneCon)*, Dec. 2019, pp. 1-5, doi: 10.1109/PuneCon46936.2019.9105847.

[3] R. S. Al-Gharibi, "IoT-based hydroponic system," in *Proc. Int. Conf. Syst., Comput., Autom. Netw. (ICSCAN)*, Jul. 2021, pp. 1-6, doi:10.1109/icscan53069.2021.9526391.

[4] K. Kularbphettong, U. Ampant, and N. Kongrodj, "An automated hydroponics system based on mobile application," *Int. J. Inf. Educ. Technol.*, vol. 9, no. 8, pp. 548-552, 2019, doi:10.18178/ijiet.2019.9.8.1264.

[5] H. Harniati, W. Trisnasari, and T. R. Saridewi, "Smart greenhouse technology for hydroponic farming: Is it viable and profitable business?," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 13, no. 4, pp. 1333-1341, Aug. 2023, doi: 10.18517/ijaseit.13.4.17916.

[6] E. Simanungkalit, M. Husna, and J. S. Tarigan, "Smart farming on IoT-based aeroponik systems," *Sinkron*, vol. 8, no. 1, pp. 505-511, Jan. 2023, doi: 10.33395/sinkron.v8i1.11988.

[7] M. C. A. Prabowo, A. A. Janitra, and N. M. Wibowo, "Sistem monitoring hidroponik berbasis IoT dengan sensor suhu, pH, dan ketinggian air menggunakan ESP8266," *J. Tecnoscienza*, vol. 7, no. 2, pp. 312-323, Apr. 2023, doi: 10.51158/tecnoscienza.v7i2.894.

[8] M. W. Al Azad et al., "CLEDGE: A hybrid cloud-edge computing framework over information centric networking," in *Proc. IEEE 46th Conf. Local Comput. Netw. (LCN)*, Oct. 2021, pp. 589-596, doi:10.1109/LCN52139.2021.9525006.

[9] Q. Qi and F. Tao, "A smart manufacturing service system based on edge computing, fog computing, and cloud computing," *IEEE Access*, vol. 7, pp. 86769-86777, 2019, doi: 10.1109/ACCESS.2019.2923610.

[10] T. Wang et al., "Fog-based computing and storage offloading for data synchronization in IoT," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4272-4282, Jun. 2019, doi: 10.1109/JIOT.2018.2875915.

[11] F. Carpio, M. Michalke, and A. Jukan, "Engineering and experimentally benchmarking a serverless edge computing system," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2021, pp. 1-6, doi: 10.1109/globecom46510.2021.9685235.

[12] A. M. Shanshool and N. A. F. Abbas, "A fog computing framework in IoT healthcare environment: Towards a new method based on tasks significance," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 12, no. 6, pp. 2274-2281, Nov. 2022, doi: 10.18517/ijaseit.12.6.16047.

[13] K. Geihs, H. Baraki, and A. de la Oliva, "Performance analysis of edge-fog-cloud architectures in the Internet of Things," in *Proc. IEEE/ACM 13th Int. Conf. Util. Cloud Comput. (UCC)*, Dec. 2020, pp. 374-379, doi: 10.1109/UCC48980.2020.00059.

[14] T. Rahman and S. K. Paul, "A review of computational tools, techniques, and methods for sustainable supply chains," in *Computational Intelligence Techniques for Sustainable Supply Chain Management*. Elsevier, 2024, pp. 1-26, doi: 10.1016/B978-0-443-18464-2.00008-X.

[15] S. Singh, "Optimize cloud computations using edge computing," in *Proc. Int. Conf. Big Data, IoT Data Sci. (BID)*, Dec. 2017, pp. 49-53, doi: 10.1109/BID.2017.8336572.

[16] Y. Jararweh et al., "The future of mobile cloud computing: Integrating cloudlets and mobile edge computing," in *Proc. 23rd Int. Conf. Telecommun. (ICT)*, May 2016, pp. 1-5, doi:10.1109/ICT.2016.7500486.

[17] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An overview on edge computing research," *IEEE Access*, vol. 8, pp. 85714-85728, 2020, doi: 10.1109/access.2020.2991734.

[18] J. Ren et al., "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031-5044, May 2019, doi: 10.1109/tvt.2019.2904244.

[19] W. Shi, G. Pallis, and Z. Xu, "Edge computing [Scanning the issue]," *Proc. IEEE*, vol. 107, no. 8, pp. 1474-1481, Aug. 2019, doi:10.1109/jproc.2019.2928287.

[20] L. U. Khan et al., "Edge-computing-enabled smart cities: A comprehensive survey," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10200-10232, Oct. 2020, doi: 10.1109/jiot.2020.2987070.

[21] J. Zhang et al., "Development of an edge computing-based cyber-physical machine tool," *Robot. Comput.-Integr. Manuf.*, vol. 67, Feb. 2021, doi: 10.1016/j.rcim.2020.102042.

[22] M. Wang and Q. Zhang, "Optimized data storage algorithm of IoT based on cloud computing in distributed system," *Comput. Commun.*, vol. 157, pp. 124-131, May 2020, doi:10.1016/j.comcom.2020.04.023.

[23] T. Diaby and B. B. Rad, "Cloud computing: A review of the concepts and deployment models," *Int. J. Inf. Technol. Comput. Sci.*, vol. 9, no. 6, pp. 50-58, Jun. 2017, doi: 10.5815/ijitcs.2017.06.07.

[24] S. Koehler et al., "Real world applications of cloud computing: Architecture, reasons for using and challenges," *Asia Pac. J. Energy Environ.*, vol. 7, no. 2, pp. 93-102, Dec. 2020, doi:10.18034/apjee.v7i2.698.

[25] C. T. Kamanga, E. Bugingo, S. N. Badibanga, E. M. Mukendi, and O. Habimana, "Cost-effective and low-complexity non-constrained workflow scheduling for cloud computing environment," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 13, no. 1, pp. 371-379, Jan. 2023, doi:10.18517/ijaseit.13.1.17752.

[26] A. Wittig and M. Wittig, *Amazon Web Services in Action*, 3rd ed. Shelter Island, NY: Manning, 2023.

[27] D. V. Kornienko, S. V. Mishina, S. V. Shcherbatykh, and M. O. Melnikov, "Principles of securing RESTful API web services developed with python frameworks," *J. Phys.: Conf. Ser.*, vol. 2094, no. 3, Nov. 2021, doi: 10.1088/1742-6596/2094/3/032016.

[28] O. Zimmermann, M. Stocker, D. Lübke, U. Zdun, and C. Pautasso, *Patterns for API Design: Simplifying Integration with Loosely Coupled Message Exchanges*, 1st ed. Stuttgart, Germany: Addison-Wesley, 2022.

[29] R. Koçi, X. Franch, P. Jovanovic, and A. Abelló, "Web API evolution patterns: A usage-driven approach," *J. Syst. Softw.*, vol. 198, Apr. 2023, doi: 10.1016/j.jss.2023.111609.

[30] G. Brito and M. T. Valente, "REST vs GraphQL: A controlled experiment," in *Proc. IEEE Int. Conf. Softw. Archit. (ICSA)*, Mar. 2020, doi: 10.1109/ICSA47634.2020.00016.

[31] A. A. Mustofa, Y. A. Dagnew, P. Gantela, and M. J. Idrisi, "SECHA: A smart energy-efficient and cost-effective home automation system for developing countries," *J. Comput. Netw. Commun.*, vol. 2023, pp. 1-12, Mar. 2023, doi: 10.1155/2023/8571506.

[32] X. Xiong, C. Wu, B. Hu, D. Pan, and F. Blaabjerg, "Transient damping method for improving the synchronization stability of virtual synchronous generators," *IEEE Trans. Power Electron.*, vol. 36, no. 7, pp. 7820-7831, Jul. 2021, doi: 10.1109/TPEL.2020.3046462.